Avaliação de consumo de potência no processador MIPS

Raphael Rocha da Silva[1], **Riccieli Minakawa**[1],
Lucas Previtali Marin[1], **João Victor Roman**[1]

[1]Faculdade de Computação – Universidade Federal de Mato Grosso do Sul
Campo Grande, MS, Brasil

Agosto de 2016

Resumo. *Este trabalho contém os resultados da estimativa de consumo dinâmico do MIPS rodando aplicações em plataformas multi-core e da energia consumida na execução de cada aplicação para diferentes quantidades de núcleos. Os resultados apresentados foram obtidos a partir das seguintes aplicações:* Stringsearch*,* Sha *e* Basicmath*. Cada uma delas foi simulada com diferentes quantidades de núcleos. A quantidade de núcleos utilizados para simulação foram 1, 2, 4, 8 e 16. A interconexão de rede (que não tem tanto impacto no consumo potência) foi fixada como* Router-LT*. Este trabalho foi realizado na disciplina "Tópicos em Arquitetura de Computadores", sob a orientação da professora Liana Duenha.*

1. Introdução

Serão apresentados nesta seção de forma sucinta alguns conceitos importantes que foram aplicados nesse trabalho.

1.1. Potência, Energia e Trabalho

Os conceitos de energia, potência e trabalho são utilizados em diferentes contextos quando relacionados a sistemas de computadores. O **trabalho** envolve tarefas diretamente associadas à execução de programas (por exemplo adição, subtração e acesso ou escrita na memória). Já a **potência** é a taxa a qual o sistema consome energia elétrica enquanto realiza essas atividades e a **energia** é a quantidade total de energia elétrica que o sistema consome no decorrer de um período de tempo. A potência é mensurada em watts enquanto a energia é medida em joules.

Realizar melhorias no desempenho de um processador, como por exemplo, aumentar o número de *transistores* na área ocupada pelo chip ou melhorar a *performance* do processador gera um acréscimo no consumo de potência, o que acarreta em um aumento de dissipação de calor, dificulta o resfriamento e a expansão do circuito, diminui a autonomia dos dispositivos além de dificultar a proximidade de componentes[3].

Outro fator que deve ser levado em conta é em relação à adição de novos hardwares no chip que acaba elevando o consumo de potência nos sistemas. Isso além de aumentar a dissipação de calor no processador, diminui significativamente a vida útil dos componentes e eleva os custos com energia

Nesse sentido, é importante estimar o consumo de potência de um circuito durante seu projeto, pois, em computação de alto desempenho, o consumo de potência tem

influência direta no número de transistores que podem ser integrados em uma mesma pastilha de silício além de ser um fator limitante em relação à frequência de operação do clock do sistema[5][4].

Vale citar que reduzir o consumo de potência nem sempre implica em reduzir o consumo de energia. Quando a frequência de clock é reduzida, por exemplo, há uma redução na potência dissipada, mas o mesmo trabalho computacional levará mais tempo para ser realizado, o que manterá o mesmo consumo de energia ao final da realização da atividade

1.2. Consumo de Potência

O consumo de potência de um circuito é geralmente determinado primariamente por uma **potência estática** dissipada por correntes de fuga e pela **potência dinâmica**, que é constituída por uma dissipação de potência de curto-circuito e pela potência de chaveamento consumida durante o carregamento ou descarregamento de cargas de capacitância[6].

A parcela dinâmica é a principal fonte de dissipação de potência na grande maioria dos circuitos[2], sendo quadraticamente proporcional à voltagem fornecida. Ela é composta pela potência de chaveamento (*switching power*) e pela potência de curto circuito (*short-circuit power*).

2. Metodologia

Estão descritos a seguir os experimentos feitos, listando as ferramentas utilizadas e os métodos escolhidos para trabalhar com elas.

2.1. Ferramentas

Para alcançar os objetivos deste trabalho, foi necessário utilizar uma ferramenta que permite simular a execução de aplicações em processadores e que permite alterar as configurações desses processadores. Para isso, foi usado o **MPSocBench**[1], um conjunto de simuladores de processadores desenvolvido na Universidade de Campinas que permite a personalização das configurações do processador simulado, e que inclui também aplicações para benchmark retiradas dos pacotes MiBench.

2.2. Aplicações

Foram usadas as seguintes aplicações de benchmark, todas elas inclusas por padrão no MPSocBench.

- **StringSearch:** Algoritmo que faz uma busca em uma cadeia de caracteres.
- **BasicMath:** Realiza operações matemáticas para as quais não é costume haver um hardware dedicado, como resolução de funções cúbicas, raiz quadrada de números inteiros e conversão de ângulos de graus para radianos. O benchmark foi escrito com o intuito de simular operações frequentemente realizadas para calcular velocidade de veículos em pistas e outras aplicações onde se precise medir velocidade indiretamente.
- **SHA:** Do inglês *Secure Hash Algorithm* (algoritmo de dispersão seguro), consiste em operações matemáticas para computar um hash. Pode ser usado para a troca de chaves de criptografia. É usado também nas funções MD4 e MD5. A entrada é formada por cadeias de caracteres longas e curtas, extraídas de um texto qualquer. Produz como saída uma mensagem de 160 bits.

2.3. Ambiente

Todas as simulações foram feitas utilizando um computador pessoal com as seguintes configurações:

- Sistema Operacional: Linux Falcom 2.0 Cinnamon 64 bit
- Kernel do Linux: 3.19.0 32 generic
- Processador: AMD AthlonTM II X2 B26 Processor x 2
- Memória: 1.7 GiB
- Disco rígido: 162.1 GB

2.4. Experimentos

Os experimentos foram feitos por meio de simulações onde fixou-se o processador como sendo de arquitetura MIPS e variou-se o número de núcleos entre 1, 2, 4, 8, 16. Para cada quantidade de núcleo, foi executada uma aplicação de cada vez para obter-se o relatório de energia referente ao processador e seus núcleos.

Nos experimentos obtém-se o consumo total e energia e potência para cada configuração de processador. Também obtém-se o consumo de cada núcleo por milissegundo. Para este último, foi necessário a alteração no tamanho da janela de amostragem, por conta do número diferente necessário de ciclos que cada aplicação necessita para executar.

Definiu-se aproximadamente 100 a 300 amostras, para isso alterou-se o tamanho da janela entre 10000, 50000, 70000, 80000, 100000, 400000 e 800000 ciclos. Tamanho da janela é o intervalo de amostragem de energia definido pela quantidade de ciclos. Na mudança de número de núcleos foi necessário essa alteração também, já que isso altera o número de ciclos para a aplicação.

3. Resultados

Quando executados com um único núcleo, os três aplicativos, BasicMath, SHA e StringSearch, tiveram um processamento com comportamento muito similar: o processador trabalhou com potência máxima do começo ao fim da execução. Isso pode ser visto nas Figuras 1, 2 e 3. Note que as escalas de tempo são diferentes de um gráfico para outro.

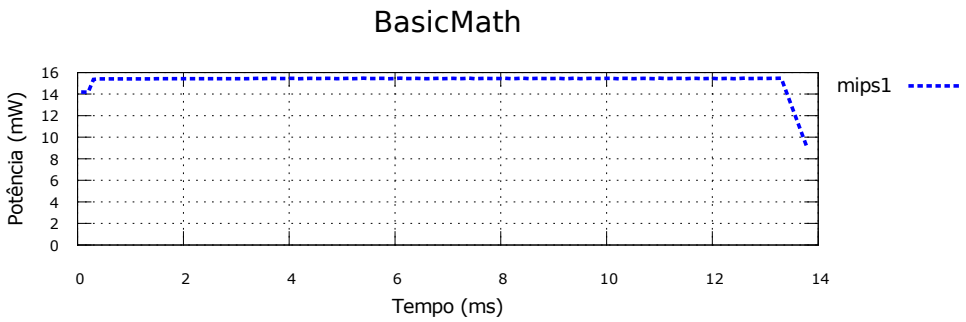

Figura 1. Linha do tempo da execução simulada do BasicMath com um núcleo.

A simulação do BasicMath quando executada com dois núcleos mostrou os dois processadores trabalhando com potência máxima pela mesma quantidade de tempo do

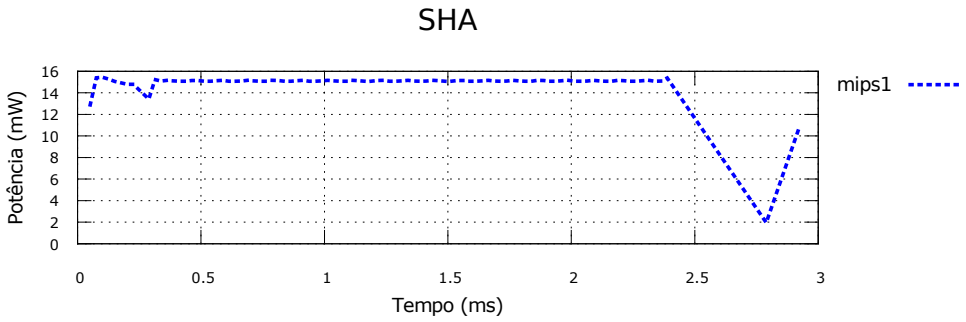

Figura 2. Linha do tempo da execução simulada do SHA com um núcleo.

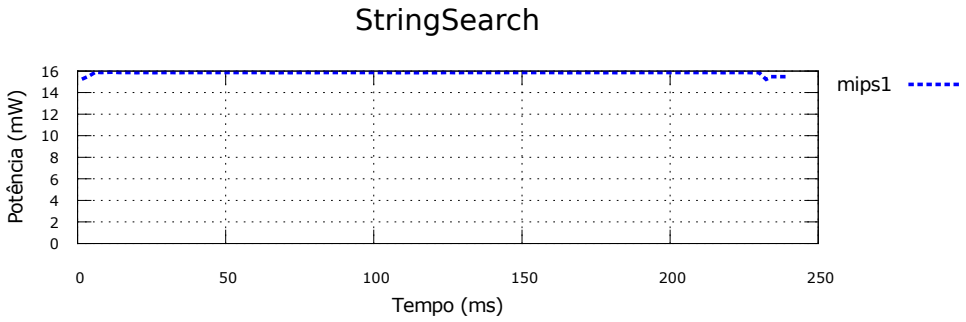

Figura 3. Linha do tempo da execução simulada do StringSearch com um núcleo.

que quando foi executada com um único núcleo, como mostra o gráfico da Figura 4. Isso sugere que esses dois núcleos estavam realizando trabalhos redundantes. Depois desse tempo executando em potência máxima, um dos processadores passou a executar com potência mínima, aguardando o outro finalizar. O processo demorou cerca de três vezes mais para concluir do que a execução com um único núcleo.

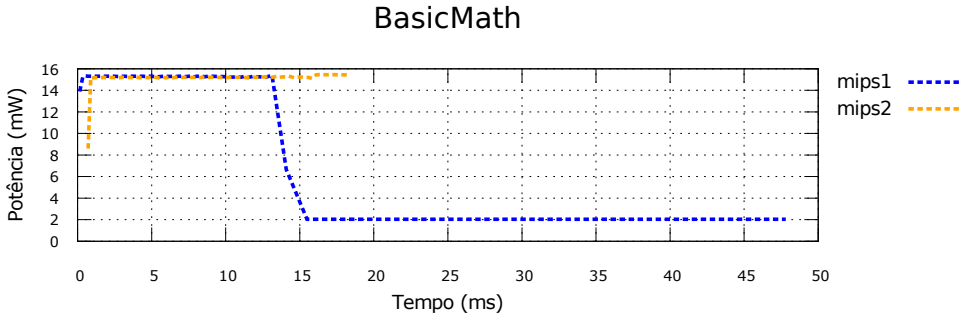

Figura 4. Linha do tempo da execução simulada do BasicMath com dois núcleos.

Similar situação ocorreu com o BasicMath executado com quatro núcleos, mostrada na Figura 5. Os quatro executaram em potência máxima pela mesma quantidade de tempo que o ocorrido com dois e com um núcleo. O tempo total de execução ficou maior do que com um núcleo, mas menor do que com dois.

O SHA com dois núcleos mostra que um dos núcleos trabalhou continuamente em potência máxima durante $3\,\mathrm{ms}$, mesmo tempo que o processador com um núcleo demorou

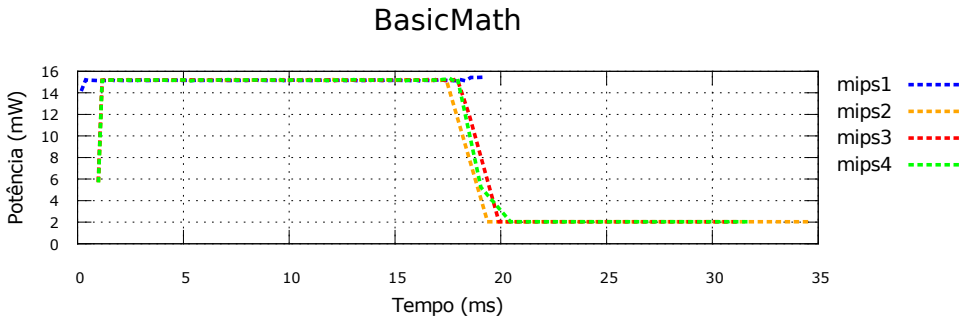

Figura 5. Linha do tempo da execução simulada do BasicMath com quatro núcleos.

para finalizar a execução. O tempo que esses dois núcleos levaram para sincronizar e finalmente encerrar a execução foi 30 vezes maior do que o processador com um núcleo. A execução com quatro núcleos mostra um atraso para começar a execução (momento em que um dos núcleos trabalha com potência máxima) do que a execução com quatro núcleos, porém a finalização ocorreu mais rapidamente.

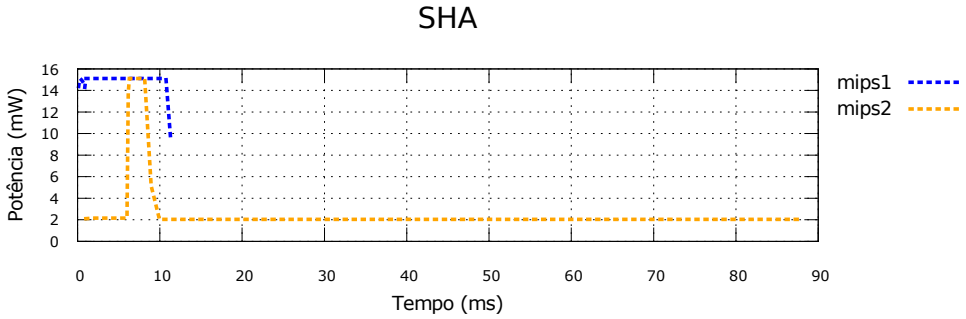

Figura 6. Linha do tempo da execução simulada do SHA com dois núcleos.

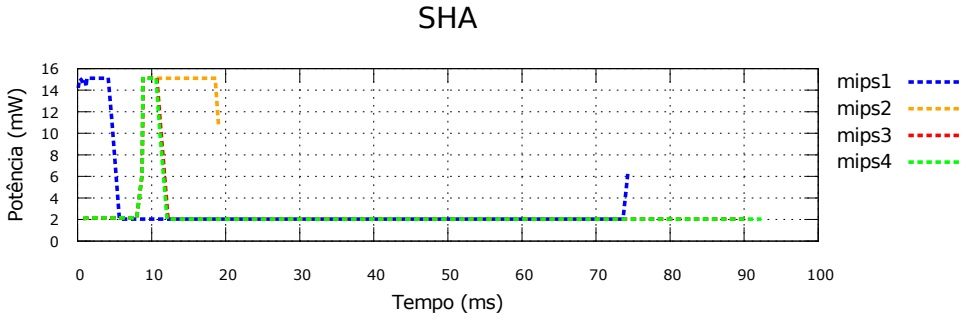

Figura 7. Linha do tempo da execução simulada do SHA com quatro núcleos.

Das três aplicações, a StringSearch foi a única que apresentou diminuição do tempo de execução em função do aumento da quantidade de núcleos. Com dois núcleos, o tempo em que os dois passaram executando à potência máxima foi próximo da metade do tempo que um núcleo executou. Isso se repetiu com quatro núcleos: o tempo em potência máxima foi metade do que quando se executou com dois núcleos. Isso sugere

que o algoritmo tem uma alta taxa de paralelização. Note ainda que tempos adicionais foram gastos para começar ou finalizar a execução. Esse tempo pode ser visto nos gráficos quando a potência dos núcleos fica baixa, próxima a $2\,\mathrm{W}$. Isso faz com que o tempo total de execução seja maior do que a metade quando se dobra a quantidade de núcleos.

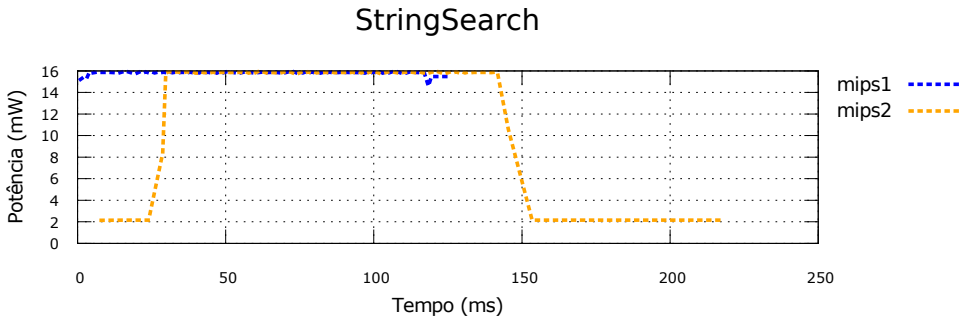

Figura 8. Linha do tempo da execução simulada do StringSearch com dois núcleos.

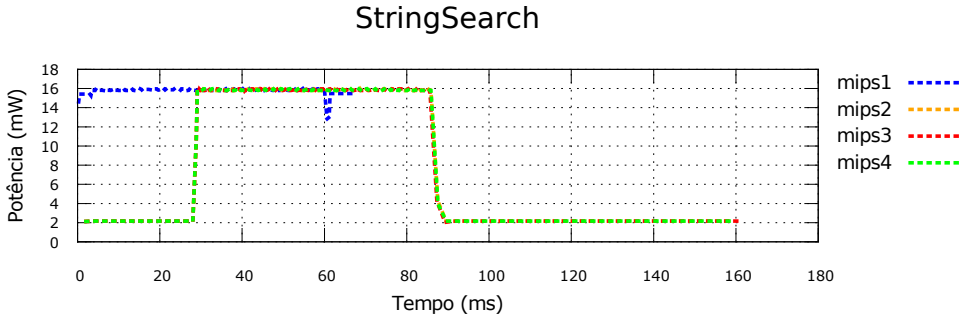

Figura 9. Linha do tempo da execução simulada do StringSearch com quatro núcleos.

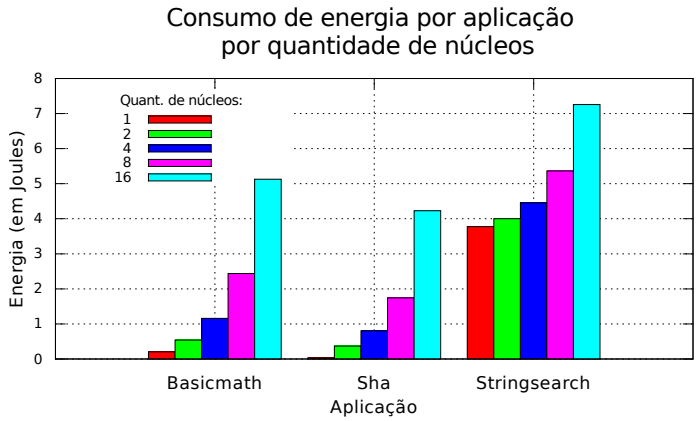

Figura 10. Comparação do consumo de energia pelas aplicações em função da quantidade de núcleos.

A Figura 10 mostra a quantidade simulada de energia consumida para executar cada uma das aplicações de acordo com a quantidade de núcleos do processador em que

essa execução é feita. A aplicação BasicMath apresentou um gasto de energia duas vezes maior cada vez que a quantidade de núcleos foi dobrada. A aplicação SHA apresentou um aumento maior do que isso, e quando passou de um para dois núcleos esse consumo cresceu mais de 100 vezes. Já a StringSearch, que foi a única aplicação a se mostrar eficiente para paralelismo, apresentou um aumento de consumo muito pequeno com o aumento do número de núcleos. Quando comparado o consumo com 1 e com 16 núcleos, esse consumo nem chegou a dobrar.

4. Conclusão

Com este trabalho, pôde-se ter uma ilustração de como aumentar o número de processadores sempre acarreta em um aumento da energia consumida. Isso acontece porque, mesmo se existisse uma aplicação que fosse totalmente paralelizável, haveria um gasto de energia só para dividir as tarefas, e depois para juntar e sincronizar as partes que cada processador teria processado.

Nota-se também claramente que cada aplicação tem um comportamento diferente em relação ao aumento de consumo de energia em decorrência do aumento de núcleos: quanto mais paralelizável é a aplicação, menor se mostra o aumento do gasto de energia.

Quanto ao tempo de execução, observou-se que nem sempre se obtém melhora quando se usa mais núcleos. Isso acontece porque os programas testados foram escritos para ser executados sob processadores de um único núcleo. Garantir uma execução mais rápida deve partir do código-fonte, que idealmente deveria conter informações sobre como separar várias threads que seriam executadas paralelamente[7]. Na ausência dessas informações, os programas que são escritos para serem executados sob um único núcleo tendem a ter um mau comportamento quando executados em mais de um núcleo.

Ressalta-se que o trabalho desenvolvido aqui considera apenas a potência dinâmica nos processadores. É comum que a energia estática (desconsiderada aqui) seja maior do que a dinâmica, e por vezes tão maior que faz com que a dinâmica seja desprezível.

Referências

[1] Mpsocbench. `http://www.archc.org/benchs/mpsocbench/`. Acessado em 22 de agosto de 2016.

[2] N. Chabini, E. M. Aboulhamid, and Y. Savaria. Determining schedules for reducing power consumption using multiple supply voltages. In *Proceedings 2001 IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD 2001*, pages 546–552, 2001.

[3] R. C. L. da Silva. D-Power: Ferramenta VHDL para estimar o consumo de potência dinâmica em arquiteturas superescalares. Dissertação de mestrado em Ciência da Computação, Universidade Estadual de Maringá, Centro de Tecnologia, Dep. de Informática, 2010.

[4] R. A. Gonçalves. Ferramenta para análise de consumo de potência em arquiteturas SMT. Dissertação de mestrado em Ciência da Computação, Universidade Estadual de Maringá, Maringá, 2008.

[5] M. Gowan, L. Biro, and D. Jackson. Power considerations in the design of the Alpha 21264 Microprocessor. In *Proceedings of the 35th Annual Design Automation Conference*. São Francisco, 1998.

[6] B. K. Mathew. The perception processor. Tese de doutorado em Ciência da Computação, Universidade de Utah, 2004.

[7] D. Woods. Multi-core slow down. `http://www.forbes.com/2009/11/23/google-microsoft-programming-technology-cio-network-multicore-hardware.html`, 2009. Acessado em 22 de agosto de 2016.